STIC Translation Branch Request Form for T

Phone: 308-0881 Crystal Plaza ¾, Room 2C15 http://ptoweb/patents/stic/s

PTO 2004-0037

S.T.I.C. Translations Branch

Information in shaded areas marked with an * is required
Fill out a separate Request Form for each document

*U. S. Serial No. : 09/649958

*Requester's Name: Ashok B Patel 80102 Phone No.: 305-2655

Office Location: SR05 Art Unit/Org. : AU 2127

Is this for the Board of Patent Appeals? No

Date of Request: 10/1/03

*Date Needed By: _____

(Please indicate a specific date)

Document Identification (Select One):

Note: If submitting a request for patent translation, it is not necessary to attach a copy of the document with the request.
If requesting a non-patent translation, please attach a complete, legible copy of the document to be translated to this form and submit it at your EIC or a STIC Library.

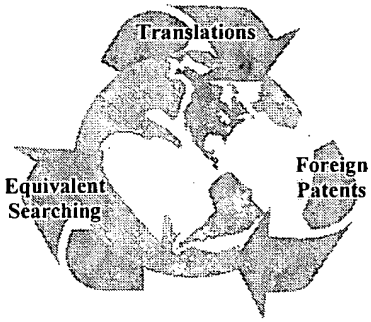
1. X **Patent** *Document No. JP 63311442 A (63-311442)
*Country Code _____
*Publication Date 12-20-88
*Language _____
No. of Pages _____ *(filled by STIC)*

2. _____ **Article** *Author _____
*Language _____
*Country _____

3. _____ **Other** *Type of Document _____
*Country _____
*Language _____

RECEIVED 2003 OCT -1 PM 2:17 TRANSLATIONS DIVISION STIC SCIENTIFIC LIBRARY

Translations Branch
The world of foreign prior art to you.



To assist us in providing the most cost effective service, please answer these questions:

- **Will you accept an English Language Equivalent?** _____ (Yes/No)
- **Would you like to review this document with a translator prior to having a complete written translation?**
  (Translator will call you to set up a mutually convenient time) _____ Yes/No)
- **Would you like a Human Assisted Machine translation?** _____ (Yes/No)
  Human Assisted Machine translations provided by Derwent/Schreiber is the default for Japanese Patents 1993 onwards with an Average 5-day turnaround

LCT in Copy E-Mail 10.14.03

STIC USE ONLY

Copy/Search
Processor: SG
Date assigned: 10/1
Date filled: 10/1
Equivalent found: (Yes/No)

Doc. No.: _____
Country: _____

Translation
Date logged in: 10.1.03
PTO estimated words: 2110
Number of pages: 8
In-House Translation Available: _____

In-House
Translator: _____
Assigned: _____
Returned: _____

Contractor: MC
Name: _____
Priority: S
Sent: 10-1-03
Returned: 10-14-03

sira Search and Information Resources Administration

Scientific Technical Information Center STIC USPTO

PAT-NO: JP363311442A

DOCUMENT-IDENTIFIER: JP 63311442 A

TITLE: MULTIPLE OPERATING SYSTEM

PUBN-DATE: December 20, 1988

INVENTOR-INFORMATION:
NAME
KUWATSURU, KEIICHIRO
SUGANO, ATSUSHI
UEDA, KENICHI

ASSIGNEE-INFORMATION:

NAME	COUNTRY
MATSUSHITA ELECTRIC IND CO LTD	N/A

APPL-NO: JP62147571

APPL-DATE: June 12, 1987

INT-CL (IPC): G06F009/46

ABSTRACT:

PURPOSE: To realize multiple functions and high-speed working with a multiple operating system OS by providing plural OSs on a single computer and adding a function to one of two OSs to utilize an interruption processing program of the other OS so that the interruption processing is ensured with the other OS with the proper functions of both OSs kept available.

CONSTITUTION: An interruption vector generator 4 gives an interruption to an OS2 for interruption in a state where the OS2 and an OS3 are executing the

processes of a task 7 and a task 8 respectively. Under such conditions, an
interruption processing call part 1 calls out an interruption processing
program 6 of the OS3 and performs the interruption processing to the OS2.
While the task 7 receives the interruption service peculiar to the OS2.
Meanwhile the OS3, i.e., the owner of the program 6 receives the information of
the interruption processing and continues the task 8 with no stop. When the
program 6 is ended in the OS2, the OS2 carries out again the task 7 based on
the original interruption program 5. Thus coexistence is possible with both
OSs without deteriorating their processing functions with each other even with
the processing progressed by interruptions.

⑲日本国特許庁（JP）　⑪特許出願公開

⑫公開特許公報（A）　昭63－311442

⑤Int.Cl.⁴	識別記号	庁内整理番号	㊸公開　昭和63年(1988)12月20日
G 06 F 9/46	350	7056－5B	

審査請求　未請求　発明の数 1　（全4頁）

㊹発明の名称　マルチオペレーテイングシステム

㉑特　願　昭62－147571
㉒出　願　昭62(1987)6月12日

㉒発明者　桑鶴　敬一郎　神奈川県川崎市多摩区東三田3丁目10番1号　松下技研株式会社内
㉒発明者　菅野　淳　神奈川県川崎市多摩区東三田3丁目10番1号　松下技研株式会社内
㉒発明者　上田　謙一　神奈川県川崎市多摩区東三田3丁目10番1号　松下技研株式会社内
㉑出願人　松下電器産業株式会社　大阪府門真市大字門真1006番地
㉔代理人　弁理士　中尾　敏男　外1名

明　細　書

1. 発明の名称

マルチオペレーティングシステム

2. 特許請求の範囲

複数のオペレーティングシステムを備え、1つのオペレーティングシステムの割込処理プログラムの中に他のオペレーティングシステムの割込処理プログラムを利用する手段を設けたマルチオペレーティングシステム。

3. 発明の詳細な説明

産業上の利用分野

本発明は、複数のオペレーティングシステム（以下OSと記す）が共存するいわゆるマルチオペレーティングシステムに関するものである。

従来の技術

最近、コンピュータが高性能化するにつれて、複数のプログラム間を切換えながら複数のタスクを行わせるマルチタスクオペレーティングシステムが実用化されている。このマルチタスクオペレーティングシステムは、一度に複数のタスクが行なえるため、シングルタスクオペレーティングシステムより高速化が図られるが、複数のタスク間の切換え時間が必らずしも高速ではなく、内外部の状態変化を迅速に処理するには十分とは云えない。このため、複数のOSを共存させるマルチオペレーティングシステムが考えられている。

発明が解決しようとする問題点

しかし、複数のOSが共存すると、その中の1つのOSのみがタイマ、コンソール入出力管理等の割込み処理ができなくなる。すなわち、各OS特有のタイマ割込みによる処理、コンソールからの割込みによる処理が行えなくなるという問題があった。本発明は以上のような従来の欠点を除去するものであり、簡単な構成で各OS独自のタイマ管理、コンソール入出力管理を生かすことができ、高機能化、高速化を可能としたマルチオペレーティングシステムを提供することを目的とするものである。

問題点を解決するための手段

上記目的を達成するために、本発明は複数のOS

を備え、1つのOSの割込処理プログラムの中に他のOSの割込処理プログラムを呼出してとり込むようにしたものである。

作　　用

上記構成において、複数のOSがそれぞれ独立してタスクを実行し、1つのOSにおいて他のOSの割込処理プログラムを実行したいときはその割込処理プログラムを呼出してとり込むことにより割込処理し、かつその間割込処理プログラムを呼出されたOSはタスクを中断することなくそのまま継続する。

実　施　例

以下、本発明の実施例について図面とともに詳細に説明する。

第1図において、1は割込処理呼出部で、割込みにより割込み処理ルーチンを起動する割込みベクタ発生器で起動されてOS3の割込処理プログラム6を実行後OS2の割込処理プログラム5にもどる一連のルーチンを行う。7はOS2のタスク、8はOS3のタスクである。

OS2がタスク7におけるプロセスを、OS3がタスク8のプロセスを実行している状態において、割込みベクタ発生器4がOS2に対して割込みを指示すると、割込処理呼出部1はOS3の割込み処理プログラム6を呼出してOS2に対して割込み処理を行い、タスク7はOS2独自の割込みサービスを受ける。この間割込み処理プログラム6の所有側であるOS3は割込処理の通知を受けるが停止することはなく、タスク8を継続する。OS2において割込み処理プログラム6が終了するとOS2は再びもとの割込処理プログラム5にもとづいたタスク7を実行する。このように、本発明によれば割込みにより進行する処理に関しても、お互いに相手の処理機能を損うことのないOS同志の共存が可能となる。

第2図は本発明のマルチオペレーティングシステムにおける一実施例の概略構成図である。210，211は各々タイマ割込み処理プログラム212，213とコンソール割込み処理プログラム214，215を持ったタイマ管理、コンソール入出力管理の各機能のあるOSであり、OS 210は初期処理プログラム201、タスク202，203…を持ち、OS 211はタスク204，205…をもっている。また、OS 210はOS 211の1つのタスクとして管理される。すなわち、OS 210の初期化プログラム201をOS 211の下で起動すると、OS 211の管理により、タスク202，203…はOS 211の世界で処理を行なうので、OS 211としては、これらは1つのタスクになる。

また、220は割込ベクタ発生器である。

このような構成のもとで、OS 211がまず立上がり、初期化プログラム201を起動してOS 210を立上げるが、この時初期化プログラム201は、割込ベクタ発生器220を次のように初期化する。すなわち、タイマ割込みによりOS 210に所属するタイマ割込み処理プログラム212を起動し、コンソール入出力割込みにより同様にOS 210に所属するコンソール割込み処理プログラム214を起動するように、割込みベクタ発生器220を初期化する。一方、コンソール割込処理プログラム214を第3図、タイマ割込み処理プログラム212を第4図のように作成しておく。

このような構成のとき、例えばコンソール入出力管理の例として円（¥）とドル（$）を分割出力する場合について説明する。いま、OS 210を割込みを呼出す側のOS，OS 211を割込みが呼ばれる側のOSとする。第3図に示すように、コンソール割込処理プログラム214において割込処理宣言がされると、OS 210にコンソール割込みが通知され、識別コード、この場合は円（¥）かドル（$）かがチェックされる。識別コードが円の場合はそれがコンソール出力され、一方識別コードがドルの場合はOS 211に対し割込処理宣言がなされる。OS 211ではコンソール割込が通知されると所属するコンソール割込処理ルーチン215に従ってコンソール出力し、割込処理が終了すると割込前の状態にもどるとともにOS 210に割込処理終了が通知され、OS 210も割込前の状態にもどる。

このとき、コンソール画面を各OS毎に分割し、

第5図のように画面をウインドウ51，52に分割したい場合、OS 210からは"¥AAA¥"、OS 211からは"BBB"のように、タスク内で文字列の前段に識別子¥，$を付加したものをコンソールに出力すればよい。

次にタイマ管理の例として、タスク202を10秒間時間待ち、OS 210全体を1秒間隔の周期起動させる場合を第4図により説明する。

タスク202が処理開始により10秒の待ちに入いると、OS 210の機能によりタスク203の処理が開始する。タスク203の開始時点から1秒経過すると、OS 211に所属するタイマ割込処理プログラム213が呼出され、その働きによりOS 211のタイマ機能が作用し、OS 210全体は処理中断され、OS 211のタスク204が起動される。タスク204の開始後1秒たつと、OS 210に所属するタイマ割込処理プログラム212に復帰し、その働きによりOS 210のタイマ機能が作用し、再びOS 210下のタスク203の処理が再開する。以上の交互起動の処理が以降くりかえされていき、最初の開始から10秒経過した時に、タスク202がOS 210のタイマ管理機能により時間待ちから解除され再起動される。

以上のように、上記実施例によればOSのもつ割込み処理ルーチンの中で他のOSの割込み処理ルーチンを呼出して実行するようにすることにより、同一割込みに対するOS間の割込み処理の競合を避けることができるようになるため、複数OSの並列稼動ができるようになり、小規模計算機でも容易に多機能化が実現する。

発明の効果

以上のように本発明は1つの計算機に複数のOSを共存させ、一方のOSの機能に他方のOSの割込処理プログラムを利用する機能をもたせたもので、各OS独自の機能を生かしたまま他方のOSの割込み処理を可能とし、多機能化、高速化を図ることができる。

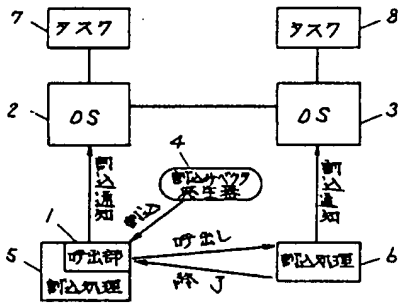
4. 図面の簡単な説明

第1図は本発明によるマルチオペレーティングシステムの全体構成を示すブロック図、第2図は、本発明によるマルチオペレーティングシステムにおけるOSが2つの場合の実施例を示す概念図、第3図は、本発明におけるコンソール割込み処理プログラムの実施例のフローチャート図、第4図は、本発明におけるタイマ割込み処理プログラムの実施例のフローチャート図、第5図は、本発明におけるマルチオペレーティングシステムのコンソール表示例を示す正面図である。

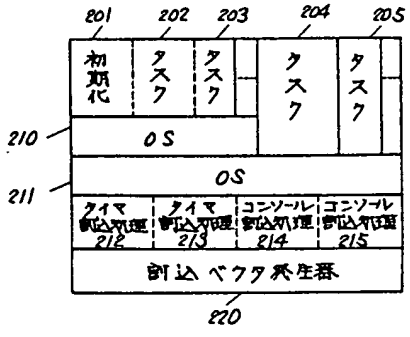
1…割込処理呼出部、2，3，210，211…OS、4，220…割込ベクタ発生器、5，6…割込処理プログラム、7，8，202～205…タスク、201…初期処理プログラム、212，213…タイマ割込処理プログラム、214，215…コンソール割込処理プログラム。

代理人の氏名　弁理士　中　尾　敏　男　ほか1名

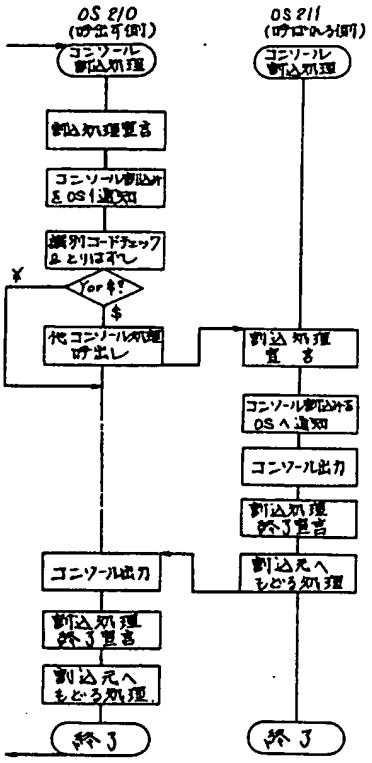
第1図



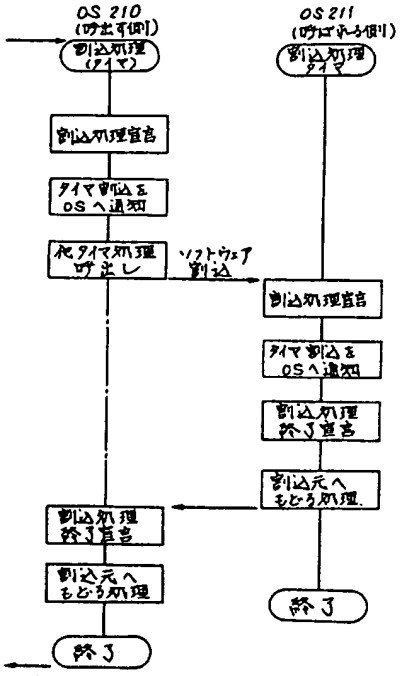
第2図



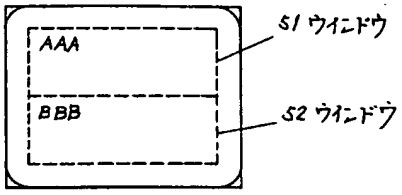
第 3 図


OS 210
(呼出す側)
OS 211
(呼ばれる側)
コンソール
割込処理
割込処理宣言
コンソール割込み
をOSに通知
識別コードチェック
&とりはずし
¥
Yor $?
$
他コンソール処理
呼出し
割込処理
宣言
コンソール割込みを
OSへ通知
コンソール出力
割込処理
終了宣言
コンソール出力
割込元へ
もどる処理
割込処理
終了宣言
割込元へ
もどる処理
終了
終了

第 4 図


OS 210
(呼出す側)
OS 211
(呼ばれる側)
割込処理
(タイマ)
割込処理
タイマ
割込処理宣言
タイマ割込を
OSへ通知
他タイマ処理
呼出し
ソフトウェア
割込
割込処理宣言
タイマ割込を
OSへ通知
割込処理
終了宣言
割込元へ
もどる処理
割込処理
終了宣言
割込元へ
もどる処理
終了
終了

第 5 図


AAA
BBB
S1 ウインドウ
S2 ウインドウ

PTO 04-37

Japanese Kokai Patent Application
No. Sho 63 [1988]-311442

MULTI-OPERATING SYSTEM

Keiichiro Kuwatsuru et al

UNITED STATES PATENT AND TRADEMARK OFFICE
WASHINGTON, D.C. OCTOBER 2003
TRANSLATED BY THE RALPH MCELROY TRANSLATION COMPANY

JAPANESE PATENT OFFICE
PATENT JOURNAL (A)
KOKAI PATENT APPLICATION NO. SHO 63 [1988]-311442

Int. Cl.[4]:	G 06 F 9/46
Sequence No. for Office Use:	7056-5B
Filing No.:	Sho 62 [1987]-147571
Filing Date:	June 12, 1987
Publication Date:	December 20, 1988
No. of Inventions:	1 (Total of 4 pages)
Examination Request:	Not filed

MULTI-OPERATING SYSTEM

[Maruchi operetingu sisutemu]

Inventor:	Keiichiro Kuwatsuru
Applicant:	Matsushita Electric Ind. Co. Ltd.

[There are no amendments to this patent.]

Claim

A multi-operating system comprising multiple operating systems, wherein in the interruption handling program of one operating system, a means to use the interruption handling program of another operating system is provided.

Detailed explanation of the invention

Industrial application field

The present invention relates to a so-called multi-operating system wherein multiple operating systems (hereafter called OS) coexist.

Prior art

Recently, as computers have achieved high performance, multi-task operating systems wherein multiple tasks are performed by switching among multiple programs have become used in practice. As multiple tasks can be concurrently performed in said multi-task operating systems, the speed can be expected to be higher compared with a single task operating system. However, switching time among multiple tasks is not always fast; thus the systems are not good enough to promptly process changing of internal and external states. Thus, multi-operating systems wherein multiple OSs coexist are considered.

Problems to be solved by the invention

However, when multiple OSs coexist, it is not possible for only one of them to perform interruption handling such as timing, console input output control or the like. That is, there is a problem that processing by timer interruption specific to each OS or processing by interruption from the console could not be handled. The present invention eliminates the aforementioned drawbacks of the prior art. The object is to provide a multi-operating system in a simple constitution, which can utilize the original timer control of each OS and console input output control, and can realize high functionality and high speed.

Means to solve the problems

To realize the aforementioned objective, the present invention comprises multiple OSs; and in the interruption handling program of one OS, the interruption handling program of another OS is called and incorporated.

Operation

In the aforementioned constitution, each of the multiple OSs independently executes its own tasks; and when an interruption handling program of another OS is desired to be executed in one OS, the interruption handling program is called and incorporated, and thereby interruption handling is performed. Besides, the OS from which the interruption handling program was called out does not interrupt its tasks during the time but continues working.

Application examples

Next, application examples of the present invention will be explained in detail in conduction with drawings.

In Figure 1, (1) indicates an interruption handling calling part, which performs a series of routines wherein it is started by an interruption vector generator that starts an interruption handling routine by interruption; and after executing interruption handling program (6) of OS

(3), returns to interruption handling program (5) of OS (2). (7) indicates the task of OS (2); while (8) indicates the task of OS (3).

While OS (3) is executing the processing of task (8), if an interruption vector generator (4) instructs interruption to OS (2) of OS (2) processing in task (7), interruption handling calling part (1) will call interruption handling program (6) of OS (3) and apply interruption handling to OS (2); and task (7) will receive OS (2)'s original interruption service. During this time, OS (3), which is the owning side of interruption handling program (6), will receive notification of interruption handling but will not stop operation but continue task (8). When interruption handling program (6) has been completed in OS (2), OS (2) will again execute task (7) based on original interruption handling program (5). In this manner, according to the present invention, even for processes that proceed by interruption, coexistence of OSs without impairing the processing function of one another will be possible.

Figure 2 is a schematic block diagram of an application example in the multi-operating system of the present invention. (210) and (211) indicate OSs with the functions of timer control and console input output control having respective timer interruption handling programs (212) and (213) and console interruption handling programs (214) and (215). OS (210) has initial processing program (201), and tasks (202), (203) ... while OS (211) has tasks (204), (205) ... Further, OS (210) is controlled as one of the tasks of OS (211). That is, if the initialization program (201) of OS (210) is started under OS (211), by the control of OS (211), tasks (202), (203) ... will be processed within OS (211); therefore, these will be one task for OS (211).

Further, (220) indicates an interruption vector generator.

Under this configuration, OS (211) will be first booted up, starting initialization program (201) to boot up OS (210). At this time, initialization program (201) initializes interruption vector generator (220) as follows. That is, interruption vector generator (220) is initialized so that timer interruption handling program (212) belonging to OS (210) will be started by timer interruption, and so that console interruption handling program (214) belonging to OS (210) will similarly be started by console input output interruption. Meanwhile, console interruption handling program (214) is prepared as illustrated in Figure 3; and timer interruption handling program (212) is prepared as illustrated in Figure 4.

With this configuration, for instance, let us explain a case as an example of console input output control wherein the Yen (¥) and the Dollar ($) are output by partitioning. Assume that OS (210) is the OS of the side that calls interruption, and OS (211) is the OS of the side where an interruption is called. As illustrated in Figure 3, when an interruption handling announcement is made in console interruption handling program (214), the console interruption is notified to OS (210), and the identification code, in this case, whether Yen (¥) or Dollar ($), will be checked. If the identification code is Yen, that will be console outputted; on the other hand, if the

identification code is Dollar, an interruption handling announcement will be made to OS (211). In OS (211), when the console interruption is notified, a console output will be made according to belonging console interruption handling routine (215); and when the interruption handling has been completed, the state before the interruption will be returned to and the completion of the interruption handling to OS (210) will be notified; thus, OS (210) will also return to the state before the interruption.

At this time, if it is preferable to divide the console screen according to each OS, and to divide the screen into windows (51) and (52) as illustrated in Figure 5, character strings with the identification element (¥ or $) within the task may be outputted to the console such as "¥AAA¥" from OS (210), and "BBB" from OS (211).

Next, as an example of timer control, a case will be explained according to Figure 4 wherein task (202) has a waiting time of 10 seconds; and all of OS (210) is periodically started with an interval of one second.

When the processing starts and task (202) enters the waiting for 10 seconds, the processing of task (203) will start by functioning of OS (210). When one second has passed since the starting of task (203), timer interruption handling program (213) belonging to OS (211) will be called; and by means of its functioning, the timer function of OS (211) will work; and all of OS (210) will have processing interrupted; and task (204) of OS (211) will be started. After 1 second has passed since starting of task (204), timer interruption handling program (212) belonging to OS (210) will be returned to; the timer function of OS (210) will work by means of its functioning; and the processing of task (203) under OS (210) will resume. The aforementioned alternate starting processing will be repeated thereafter; and when 10 seconds have passed since the initial starting time, task (202) will be released from standby by means of the timer control function of OS (210) and be restarted.

As described above, according to the aforementioned application example, by calling and executing an interruption handling routine of another OS in an interruption handling routine held by one OS, competition in the interruption handling between OSs for the same interruption can be avoided. As a result, multiple OSs can operate in parallel, and multi-functionality can be easily realized even on a small computer.

Effect of the invention

As mentioned above, according to the present invention, multiple OSs are allowed to coexist in one computer; and one OS is allowed to have the function of using the interruption handling program of the other OS; thereby while utilizing the original function of each OS, interruption handling of the other OS can be executed; and multi-functionality and high speed can be expected.

Brief description of the figures

Figure 1 is a block diagram illustrating the whole configuration of the multi-operating system according to the present invention; Figure 2 is a conceptual diagram illustrating an application example wherein two OSs coexist in the multi-operating system according to the present invention; Figure 3 is a flow chart of an application example of the console interruption handling program in the present invention; Figure 4 is a flow chart of an application example of the timer interruption handling program in the present invention; and Figure 5 is a front view illustrating an example of console display of the multi-operating system in the present invention.

1 Interruption handling calling part
2, 3, 210, 211 OS
4, 220 Interruption vector generator
5, 6 Interruption handling program
7, 8, 202 - 205 Task
201 Initial processing program
212, 213 Timer interruption handling program
214, 215 Console interruption handling program

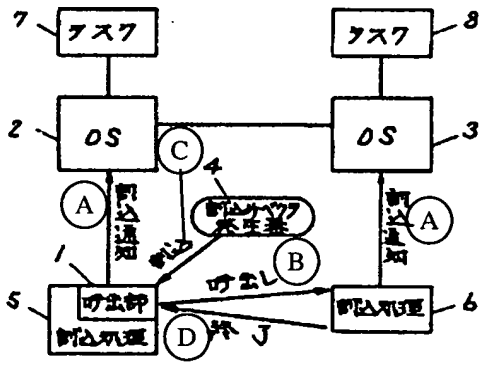


Figure 1

Key:
A Interruption notification
B Calling
C Interruption
D End
1 Calling part
4 Interruption vector generator
5,6 Interruption handling
7,8 Task

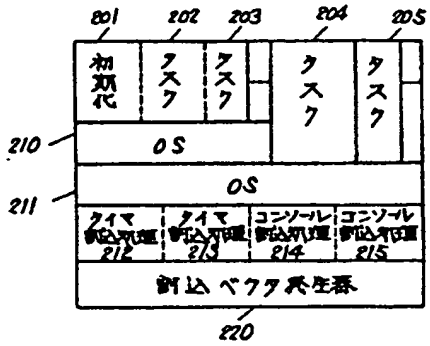


Figure 2

Key:		
	201	Initialization
	202, 203, 204, 205	Task
	212, 213	Timer interruption handling
	214, 215	Console interruption handling
	220	Interruption vector generator

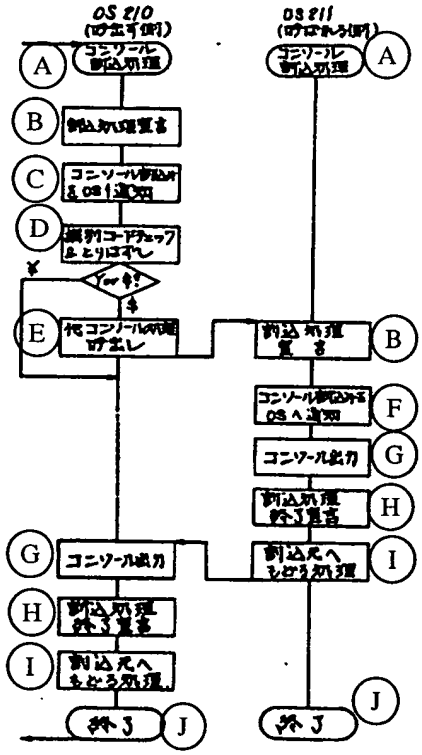


Figure 3

Key:		
	OS (2)10	Calling side
	OS (2)11	Called side
	A	Console interruption handling
	B	Interruption handling announcement
	C	Console interruption notified to OS 1

D Identification code check and removal
E Calling other console handling
F Console interruption notified to OS
G Console output
H Interruption handling ending announcement
I Processing to return to the interrupting source
J End

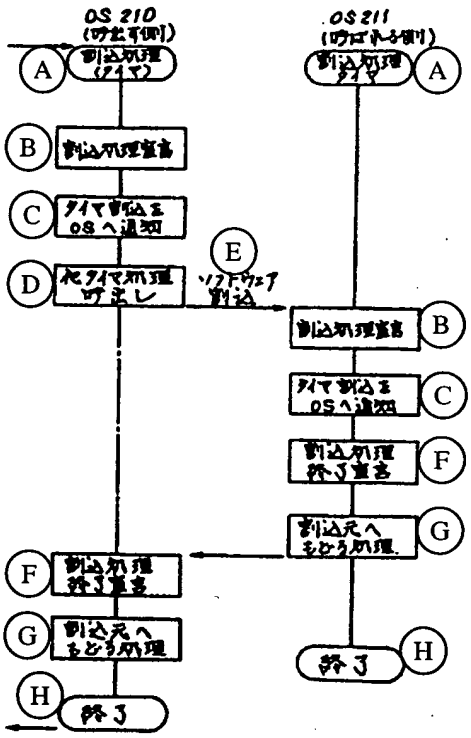


Figure 4

Key: OS (2)10 Calling side
OS (2)11 Called side
A Interruption handling (timer)
B Interruption handling announcement
C Timer interruption notified to OS
D Call other timer handling
E Software interruption
F Interruption handling ending announcement
G Processing to return to the interrupting source
H End

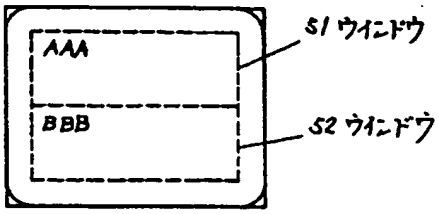


Figure 5

Key: 51, 52 Window